Naval Research Laboratory

Stennis Space Center, MS 39529-5004

NRL/MR/7320--12-9359

Environmental Measurements Path Planner (EMPath) User's Manual

Kevin D. Heaney
Richard L. Campbell
Richard H. Stroop

Ocean Acoustical Services and Instrumentation Systems, Inc.
Lexington, Massachusetts

Lucy F. Smedstad

Ocean Dynamics and Prediction Branch
Oceanography Division

Germana Peggion

University of New Orleans
New Orleans, Louisiana

June 1, 2012

REPORT DOCUMENTATION PAGE		*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 01-06-2012	**2. REPORT TYPE** Memorandum Report	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Environmental Measurements Path Planner (EMPath) User's Manual		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER 0603207N
6. AUTHOR(S) Kevin D. Heaney,* Richard L. Campbell,* Richard H. Stroop,* Lucy F. Smedstad, and Germana Peggion†		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER 73-5091-12-5
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Oceanography Division Stennis Space Center, MS 39529-5004		**8. PERFORMING ORGANIZATION REPORT NUMBER** NRL/MR/7320--12-9359
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Space & Naval Warfare Systems Command 2451 Crystal Drive Arlington, VA 22245-5200		**10. SPONSOR / MONITOR'S ACRONYM(S)** SPAWAR
		11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*Ocean Acoustic Services and Instrumentation Systems (OASIS), Inc., 5 Militia Drive, Lexington, MA 02421-4706
†University of New Orleans, 2000 Lakeshore Drive, New Orleans, Louisiana 70148

14. ABSTRACT

A suite of sensors in an oceanographic area of interest may be optimized with the use of a Genetic Algorithm. The purpose of the Environmental Measurements Path Planner (EMPath) is to use and execute a genetic algorithm to generate optimal search plans for a suite of sensors based upon constituent cost-functions (CCF) contained in an input netcdf file. EMPath has a built-in to pre-processor for environmental ocean model data to determine the largest area of model forecast uncertainty for the input file. The User's Manual discusses all possible options for the EMPath as well as required directory setups.

15. SUBJECT TERMS

Gliders	Genetic algorithm
Glider sampling	Ocean modeling

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Lucy Smedstad
a. REPORT Unclassified Unlimited	**b. ABSTRACT** Unclassified Unlimited	**c. THIS PAGE** Unclassified Unlimited	Unclassified Unlimited	19	**19b. TELEPHONE NUMBER** *(include area code)* (228) 688-5365

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Environmental Measurements Path Planner (*EMPath)* Users Manual

Purpose

The purpose of the Environmental Measurements Path Planner (*EMPath)* is to generate optimal search plans for a suite of sensors based upon constituent cost-functions (CCF) contained in an input netcdf file. There is a primary assumption, that the sensors are related to each other in function. Otherwise EMPath should be run separately for each sensor. EMPath performs two primary functions [Heaney et. al. 2007]:

1.) platform search generation
2.) genetic algorithm (GA) optimization

The CCF file contains any number of user generated cost functions either as 2D static functions: (lon,lat) or as 3D dynamic functions: (lon,lat,time). EMPath has been tested as part of the Glider Observation Strategies (GOST) Project [Smedstad et al, 2012].

Usage

EMPath is controlled by an input.prm text file and requires CCFs and bathymetry as the main input files (either in separate or the same netcdf file).

The *datacx* code (Appendix 2) is available from the default distribution to create a set of CCFs based upon the ocean spatial and temporal variability in temperature and salinity from the NRL Relocatable Nowcast/Forecast System (RELO NCOM) [Rowley, 2010]. Averages are taken over the run and over the depths.

Compilation instructions for EMPath and *datacx* are found in Appendix 3. EMPath can be run from the command line or a unix-shell with the following syntax:

```
$path/EMPath  --option1 value –option2 –option2 value2 …-option{n} value{n}
```

where $path is the path to the directory where the executable EMPath resides. Command line options supersede input.prm values.

Manuscript approved September 26, 2011.

EMPath uses two sub-directories below the execution directory. The 'data' directory is used to store individual run scores as well as the best and the output kml. The 'persistent' directory is where the morphology is stored once executed (and read from if it exists) and where the Cweight vector and Cords_initial.txt file are stored for starting the search with a pre-set parameter list for the platform.

Input Files

EMPath uses a set of required and a set of optional input files. The options can be defined in these files or from the command line.

Parameter hierarchy is: CCF netcdf file, input.prm, command line, with the latter over-writing the former. In particular, CCF weights are generally defined as 1 in the netcdf file and can be overwritten in the input.prm file. Numbers of vehicles, glider speed, morphology computation time, etc. can be adjusted from the command line, overwriting values in the input.prm file.

input.prm

The primary control file for an EMPath run is the input.prm file. This file contains parameters and filenames for the EMPath run. It is beneficial to put inputs here so that they are persistent and can be given to others to repeat results. It must reside in the directory from which the execution occurs. The following commands can be placed in the input.prm file:

(Those marked with an * are required)

* **filename** – the full path of the netCDF CCF file to read for analysis

* **bathy** – the full path of the bathy file to keep platforms from running into land

* **Bathy_Scale_Factor** – EMPath expects bathymetry in the bathy.nc file to be positive down. The Bathy_Scale_Factor can be set to -1 to invert the bathymetry. If more than 20% of a vehicles positions have run aground, a warning will be presented to the user, indicating that Bathy_Scale_Factor should be used.

runs – sets how many times to run the GA on with current settings (all of the output is saved, but do not use a set seed or all of the runs will be identical) [defaults to 1]

individuals – defines how many random permutations are in each generation (this is how to add more initial genetic material) [defaults to 200]

generations – defines how many iterations to run for one genetic evaluation (this is how many times the solution mutates before settling on a final answer) [defaults to 80]

PLATFORMS:

gliders – defines how many moving gliders to add to the search [defaults to 2]

moorings – defines the number of stationary moorings to search with [defaults to 1]

drifters – defines the number of drifting surveys [defaults to 0]

airXBTs – defines the number of airXBT surveys [defaults to 0]

shipXBT – defines the number of shipXBT surveys [defaults to 1]

glider_depth – defines maximum depth of gliders (this is given in the cost function NetCDF file but can be changed here if desired) [defaults to 400]

glider_speed – defines horizontal speed of gliders in meters per second (this is given in the cost function NetCDF file but can be changed here if desired) [defaults to 0.4 m/s]

shipXBT_speed – defines speed of shipXBT [defaults to 1.25 m/s]

hours_to_search_over – total time to run the search for [defaults to time*DeltaTime (all of the time available) from the input NC file]. This must be set to a shorter than run time if not all model run fields are used. (See Appendix 2).

hours_between_glider_turns – how often a glider turns [defaults to 12 hours]

hours_between_shipXBT_turns – how often a ship turns [defaults to 12 hours]

degrees_allowed_between_platforms – distance potential in latitude and longitude degrees (keeps platforms from being too close to each other) [defaults to .5 degrees and will not go below .1 degrees]

hours_allowed_before_resampling – time potential in hours (keeps platforms from searching the same space within a certain amount of time) [defaults to 24 hours and will not go below .1 hours]

Water_Current_Scale_Factor – scales how strong the currents are [defaults to 1]

WaterSpace – set to 1 to exclude predefined areas of the water from the search, set to 0 otherwise [defaults to 0]

lat_min_bound – defines the minimum latitude to allow platforms in (this only needs to be set to shrink the search zone from all that is included in the cost function NetCDF file) [defaults to lat_min]

lat_max_bound – defines the maximum latitude to allow platforms in (this only needs to be set to shrink the search zone from all that is included in the cost function NetCDF file) [defaults to lat_max]

lon_min_bound – defines the minimum longitude to allow platforms in (this only needs to be set to shrink the search zone from all that is included in the cost function NetCDF file) [defaults to lon_min]

lon_max_bound – defines the maximum longitude to allow platforms in (this only needs to be set to shrink the search zone from all that is included in the cost function NetCDF file) [defaults to lon_max]

OpArea – permits the definition of a polygon operational area where platforms are permitted to search. This can be specified by any number of points, but must be convex, defined in an anti-clockwise order. The algorithm is most efficient if the 2nd angle is the smallest. The syntax is for a single line with a single space following the command OpArea, then comma delimited lon/lat pairs. {OpArea N,lon1,lat1,lon2,lat2,…,lonN,latN}.An example for the Mediterranean is:

```
OpArea 4,15.3658,37.7221,15.1620,36.3715,16.9,36.45,16.9,37.4
```

[Cost Function Name] – putting the name of a cost function and then a number will set the weight of that cost function to that number (generally between 0 and 1) [Cost Functions default to a weight of 1]

Comments are added to the 'input.prm' file by starting the line with a pound symbol and a space "# ".

CCF .nc file

The NetCDF files should include:

- Dimensions: lon, lat, dynamicfunc, staticfunc, WCurrents, time, where
    - dynamicfunc is the number of functions containing 4 variables (time, lat, lon, depth)
    - staticfunc is the number of variables containing 3 variables (lat, lon, depth)
    - WCurrents is the number of water currents, this should be 2 (North-South, East-West).
- Variables: dynamicfun, staticfun, watercurrents, lat, lon, time
- global variables: GAfun_num, TotalTime, DeltaTime, Glider_Depth, Glider_Speed}.
    - GAfun_num is the number of cost functions.
    - TotalTime is the number of hours the file covers,
    - DeltaTime is the number of hours between each step.
    - Glider_Depth is how far the glider dives before surfacing, and the
    - Glider_Speed is the horizontal speed of the glider. "

An example header file from an nc file is given in Appendix 1. There are two types of cost function routines: dynamicfunc and staticfunc. The difference between them is the inclusion of a time index in dynamicfunc. The CF file requires the global variables {GAfun_num, TotalTime, DeltaTime, Glider_Depth, Glider_Speed}. Where:

GAfun_num - number of cost functions
TotalTime – number of hours in the CF functions (hrs)
DeltaTime – hours between each time index (hrs)
Glider_Depth – maximum depth extent of the dives (m)
Glider_Speed – horizontal speed of the gliders (m/s)

The names of the individual cost functions are defined in the global attributes. The Global attributes are also where the names of the CCF in dynamicfun and staticfun are defined. A matrix is either dynamic or static based on if there is a time dimension. The names must be GAfun_id# where # starts at 1 and goes until the end of the cost functions. Their contents must be in order and separated by colons. The Information must be "Name : type_of_cost_function:which_matrix_it_is_in : index_in_the_matrix : weight : a_meaningless_string_to_end_it". The Name can be anything and is just used to label the cost function. The type of cost function can be either cfun_LineAvg or cfun_LineStd based on whether the points on the platform path are to be averaged or the standard deviation found. The indices should start at 0 and go through the number of cost functions existing for the matrix. The weight will generally be 1, but may be changed depending on cost function importance. Making the weight 0 will cause it to not be evaluated and making it 2 will cause it to have twice the affect of every other cost function. The meaningless string just acts as a way to end the search and makes sure that all valuable information was found.

Bathy.nc file

The full path name of the bathy file must be in "input.prm" on a line starting with 'bathy' (the bathymetry must match or include the area defined by the cost functions). The bathymetry matrix variable must be titled ‘bathy’, ‘Bathy’, or ‘grid_water_dep’ and associating ‘lat’ and ‘lon’ variables and dimensions must be present. Using ETOPO1_Bed_g_gmt4.grd for general bathymetry will also work if it exists in the same folder as the program.

For RELO NCOM domains bathy.nc is made from
$RELO/bin/make_depth_nc.xc grid_1.nc bathy.nc

Command Line Options:

Here are some commands that can be run from the command line for ease of use and added functionality. Simply type the command followed by the desired number after it. Multiple commands may be used.

individuals, generations, gliders, moorings, drifters, airXBT, shipXBT, glider_speed, shipXBT_speed – can all be updated here but should really be set in the "input.prm" file

EMPath runs can be controlled from the command line via the following control terms. These command line run options are not input parameters and are not present in the input.prm file.

seed – this sets the seed for the current run (the same seed will return the same output)

runs – sets how many times to run the genetic algorithm on with the current settings (all of the output is saved, but do not use a set seed or all of the runs will be identical)

morph – runs the morphology to set up CWeights and the environmental picture used in plotting (if the cost functions or the input files are changed, then this should be used!)

morph_hours – changes the resolution of the morphology run (should be between 1 and the total time) . Usually morph_hours is set to 3 in the command line to permit a 3 hour glider trace. Longer glider traces have more spatial smoothing.

morph_depth – sets how deep the bathymetry must be for the search to be in bounds within morph in meters (set to 0 for land to see most platforms and set to glider depth to see where gliders can go)

setOne – uses information in "One_Indv.txt" to add a best individual into the genetic material for the evaluation

setCords – uses information in "Cords_initial.txt" to define where the platforms must start (effectively only searching bearings)

Initial Coordinates

Adding 'setCords' to the command line allows for fixed initial positions for multiple platforms. Bearings may be set for gliders and ships. The text file 'Cords_initial' that is inside of the 'persistent' folder must be edited in order to circumvent the global genetic algorithm search. The numbers of platforms to be changed must be determined as well as the order of arrival. The order of platforms is Moorings, Drifters, AirXBTs, Gliders, and ShipXBTs. For example, 1 mooring, 2 drifters, 1 glider, and 2 shipXBTs, allows for the editing of 6 platforms. The first line of the 'Cords_initial' file would edit the mooring and the second line would edit the first drifter. To skip the editing of the second drifter, (or the third platform), write -99 on the third line. Upon continuing, the fourth line would edit the glider, and the fifth line would edit the first shipXBT. After editing, if for example, the last shipXBT does not need editing, stop instead of writing -99 for all of the last platforms. When the file ends it assumes no changes on any remaining platforms.

It is assumed that the platform time 0 corresponds to model analysis time. The values are specified one line per vehicle and in order of lon, lat, initial heading, turn, turn. If a value is specified by the user, EMPath will not search over that variable. The GA will search for all parameters that are not specified.

The format fort the Cords_initial.txt is a single line for each vehicle, with space separated floating point values.

Example:
```
lon_glider1 lat_glider1 initial heading turn1 turn2
lon_glider2 lat_glider2 initial heading turn1 turn2
-99
```

lon_glider4 lat_glider4
This defines the longitudes/latitudes of gliders 1,2, and 4 and the heading and first two turns of vehicle 1 and 2.

When an input of -99 is used, EMPath will search for all parameters defining that platform (lon, lat, heading, turn1, turn2 ...). Start by adding the latitude followed by a space and then the longitude, both in degrees and decimal degrees. Accuracy is flexible regarding decimals after the degree, but the degree minute second format is not allowed. Bearings may be added to lines that corresponding with gliders and shipXBTs. These lines contain initial latitude, initial longitude, initial bearing, and then how many degrees to add or subtract from the last bearing for as many turns as the platform takes. If the platform only takes 2 turns, then there could be up to 5 initial inputs. If a bearing is not added EMPath does the initial bearing calculation. The xy space will be searched for an initial bearing of 0-360^o followed by turns of no more than 30^o.

However if the whole path of the platform is described, it will not be allowed to evolve at all and will simply be added to the score of the other platforms. The most common application of being able to set bearings would be to just set the latitude, longitude and initial bearing for the platform and then let the genetic algorithm solve the best path.

Adding Genetic Material

Adding 'setOne' to the command line defines one genome or path for the genetic algorithm to evaluate. This will then use the path to reproduce with other genomes and mutate into possibly better solutions. This can be useful if a long run exists and a good path is known, but extension of the generations or populations is desired. Path information is available from a previous run in the respective CSV file.

You must give the 'setOne' input file all of the path information, such as starting location and bearings. Failing to define everything will leave a partially empty genome, which will evaluate to zero and be useless. The text file 'One_Indv' that is inside of the 'persistent' folder must be edited. The number and order of arrival of the platforms must be known. The order of platforms is Moorings, Drifters, AirXBTs, Gliders, and ShipXBTs. Each line in the text file corresponds to one platform. Everything takes an initial latitude and longitude. The gliders and shipXBTs take initial bearings, and bearing changes in addition, both in degrees. Separate everything with spaces, and leave nothing out. Bearing changes to add are based on the hours_to_search_over divided by hours_between_glider_turns, and similar math for the shipXBTS. Another alternative is to run the genetic algorithm once and look at a CSV file.

Water Space

Other than shrinking the outer bounds of latitude and longitude in the 'input.prm' file, boxes or circles can be set up for the genetic algorithm to avoid. These are called water spaces, and are turned on by adding the line 'WaterSpace 1' anywhere in the 'input.prm' file. The water spaces are defined in the text file 'rshapes' in the 'persistent' folder. Other comments may be written in this file as it does not read everything, but rather looks for words in the right order. As long as the line starts with the shape and continues in the right format it will be read. Current comments in the file give instructions as well. The formatting for the water spaces is:

```
shape:latitude,longitude,radius,start_time,end_time,weight;
```

Pay close attention to the delimiters. A colon follows the shape, and then all of the information is separated by comas with no spaces, and everything ends with a semi-colon. The shapes can be [circle square exponent gaussian linear]. The circle and square shape should be obvious. The exponent, gaussian and linear shapes are circles that fall off at an exponential, gaussian and linear rate as they approach the center. So the center of all the shapes is the worst place for a platform but it would be allowed at the edge. Circle and square would not allow the platforms anywhere inside of them. The start_time and end_time allow for the creation of shapes that move in time. The same shape may be created at different times in the new places, but it would work. Weight indicates how important that shape is to the genetic algorithm. A weight of 1 would make it as important as the other cost functions, but it can be made more or less important based on what the shape actually represents.

Output

EMPath product files are stored in 'persistent' and 'data' directories.

The files in the 'data' directory are listed below.

'errors.txt' – outputs any errors that the GA encountered during the run instead of putting those errors on the screen (should be empty).

'output.prm' – describes the run so users can see what was input to create the data they are viewing. This can be useful if the command line was used to change things instead of inputting everything in the 'input.prm' file.

'GA_Run#.csv' – This is a comma separated variable file that contains all of the details of the best path for a full run. Use a spreadsheet program to read it neatly formatted in a table. It has the longitude, latitude and current bearing for every platform at every time. Platforms that have no need for bearing just have a 0 placeholder. The last bearing point of a glider's path is also 0 because it is no longer going anywhere and therefore has no bearing.

'scores#.dat' – shows the generational data for each run number. Zero scores usually mean that the glider ran aground. The values of scores are dimensionless and are relative for each run's cost functions, there is no standard scale.

'#_Indv.txt' – Output genome for each run. The genome values are initial lat/lon, heading and then turns for most vehicles. This text file can be truncated and then used as input in SetCords.txt for a refined search.

'output#.png' – This is a picture of the platforms moving over the morphology. The morphology is a normalization of the multi-dimensional cost functions. The morphology also allows the user to see the cost functions and where they are focusing on an area of interest.
'output#.kml' – This is a Google Earth file for viewing the morphology and the platforms. It requires Google Earth to be installed on the host computer.

'GA_Best.csv', 'GA_Best.kml' – In addition to output files for each run, the results for the run with the lowest cost function score (the best solution) is copied into GA_Best.csv and GA_best.png.

The files in the 'persistent' directory are listed below.

'morphology.txt – This text file contains the morphology matrix, for each position (lat/lon) and each constituent cost-function. The final column of each position is the combined cost function.

'CWeight.txt' – This file stores the normalization factor and the maximum value for each constituent cost-function. These values are not dependent upon user defined weights. They are used in subsequent runs if "spaces" are used to add regions where platforms are prohibited. Currently all weights are saved into one file.

'sums.txt' – This file stores the combined morphology. These values are used to generate the morphology.png. The PNG file is erased each time and if the morphology is not run, the morphology.png is regenerated using the sums.txt.

Acknowledgements
Special thanks to Clark Rowley of the Naval Research Laboratory for providing routines to make bathymetry file from RELO NCOM grid files and to David Sitton of Qinetiq North America for the sample netCDF file.

References

[1] **Kevin D. Heaney, Glen Gawarkiewicz, Timothy F. Duda and Pierre F. J. Lermusiaux**, Non-linear Optimization of Autonomous Undersea Vehicle Sampling Strategies for Oceanographic Data-Assimilation, *Journal of Field Robotics* **Special Issue on "Underwater Vehicles"**,(2007).

[2] **Clark D. Rowley**, Validation Test Report for the RELO System, NRL Memorandum Report 7320—10-9216, (2010).

[3] **Lucy F. Smedstad, Kevin D. Heaney, Germana Peggion, Charlie N. Barron, Emanuel Coehlo**, Validation Test Report for a Genetic Algorithm in the Glider Observation STrategies (GOST 1.0) Project: Sensitivity Studies, NRL Memorandum Report 7320—12-9361, (2012).

Appendix 1 sample netCDF attributes

```
netcdf ET_cfmaps_extrap_20110201 {
dimensions:
	dynamicfunc = 12 ;
	staticfunc = 12 ;
	lon = 106 ;
	WCurrents = 2 ;
	time = 35 ;
	lat = 100 ;
variables:
	float dynamicfun(dynamicfunc, time, lat, lon) ;
		dynamicfun:_FillValue = 0.f ;
		dynamicfun:units = "cost_functions" ;
	float staticfun(staticfunc, lat, lon) ;
		staticfun:_FillValue = 0.f ;
		staticfun:units = "cost_functions" ;
	float watercurrents(WCurrents, time, lat, lon) ;
		watercurrents:_FillValue = 0.f ;
		watercurrents:units = "m/s (mean over depth)" ;
		watercurrents:missing_value = 0 ;
		watercurrents:description = "mean profile V-U velocities (0-
150m)" ;
	float lat(lat) ;
		lat:units = "degrees_north" ;
	float tau(time) ;
		tau:units = "hours starting at time in filename" ;
	float lon(lon) ;
		lon:units = "degrees_east" ;
	float time(time) ;
		time:units = "day_number" ;

// global attributes:
		:title = "NETCDF File for TOFU" ;
		:author = "David Sitton" ;
		:GAfun_id0 = "Name:Operation:CostType:Index:Weight:Useless"
;
		:GAfun_num = 24 ;
		:GAfun_id1                                              =
"Ensemble_Spread_Temp_0m:cfun_LineAvg:dynamic:0:1.0:Useless" ;
		:GAfun_id2                                              =
"Ensemble_Spread_Temp_25m:cfun_LineAvg:dynamic:1:1.0:Useless" ;
```

```
                :GAfun_id3                                                  =
"Ensemble_Spread_Temp_100m:cfun_LineAvg:dynamic:2:1.0:Useless" ;
                :GAfun_id4                                                  =
"Ensemble_Spread_Sal_0m:cfun_LineAvg:dynamic:3:1.0:Useless" ;
                :GAfun_id5                                                  =
"Ensemble_Spread_Sal_25m:cfun_LineAvg:dynamic:4:1.0:Useless" ;
                :GAfun_id6                                                  =
"Ensemble_Spread_Sal_100m:cfun_LineAvg:dynamic:5:1.0:Useless" ;
                :GAfun_id7                                                  =
"Ensemble_Mean_Temp_0m:cfun_LineStd:dynamic:6:1.0:Useless" ;
                :GAfun_id8                                                  =
"Ensemble_Mean_Temp_25m:cfun_LineStd:dynamic:7:1.0:Useless" ;
                :GAfun_id9                                                  =
"Ensemble_Mean_Temp_100m:cfun_LineStd:dynamic:8:1.0:Useless" ;
                :GAfun_id10                                                 =
"Ensemble_Mean_Sal_0m:cfun_LineStd:dynamic:9:1.0:Useless" ;
                :GAfun_id11                                                 =
"Ensemble_Mean_Sal_25m:cfun_LineStd:dynamic:10:1.0:Useless" ;
                :GAfun_id12                                                 =
"Ensemble_Mean_Sal_100m:cfun_LineStd:dynamic:11:1.0:Useless" ;
                :GAfun_id13                                                 =
"Mean_Spread_Temp_over_time_0m:cfun_LineAvg:static:0:1.0:Useless" ;
                :GAfun_id14                                                 =
"Mean_Spread_Temp_over_time_25m:cfun_LineAvg:static:1:1.0:Useless" ;
                :GAfun_id15                                                 =
"Mean_Spread_Temp_over_time_100m:cfun_LineAvg:static:2:1.0:Useless" ;
                :GAfun_id16                                                 =
"Mean_Spread_Sal_over_time_0m:cfun_LineAvg:static:3:1.0:Useless" ;
                :GAfun_id17                                                 =
"Mean_Spread_Sal_over_time_25m:cfun_LineAvg:static:4:1.0:Useless" ;
                :GAfun_id18                                                 =
"Mean_Spread_Sal_over_time_100m:cfun_LineAvg:static:5:1.0:Useless" ;
                :GAfun_id19                                                 =
"H_Gradient_Temp_over_time_0m:cfun_LineAvg:static:6:1.0:Useless" ;
                :GAfun_id20                                                 =
"H_Gradient_Temp_over_time_25m:cfun_LineAvg:static:7:1.0:Useless" ;
                :GAfun_id21                                                 =
"H_Gradient_Temp_over_time_100m:cfun_LineAvg:static:8:1.0:Useless" ;
                :GAfun_id22                                                 =
"H_Gradient_Sal_over_time_0m:cfun_LineAvg:static:9:1.0:Useless" ;
                :GAfun_id23                                                 =
"H_Gradient_Sal_over_time_25m:cfun_LineAvg:static:10:1.0:Useless" ;
                :GAfun_id24                                                 =
"H_Gradient_Sal_over_time_100m:cfun_LineAvg:static:11:1.0:Useless" ;
                :TotalTime = 99. ;
                :lon_min = 14.8000001907349 ;
                :lat_max = 37.9912109375 ;
                :DeltaTime = 3. ;
                :Glider_Depth = 150. ;
                :lon_max = 17.0891056060791 ;
                :Tavg_Depth = 41.6666666666667 ;
                :Glider_Speed = 0.5 ;
                :lat_min = 36.25 ;
}
```

Appendix 2 datacx pre-conditioner

The purpose of *datacx* is to condition a set of RELO NCOM ocean forecast fields (in netcdf format) for use as a Cost Function (CF) input file for the EMPath (Environmental Mission Path Planner) genetic algorithm code. Currently EMPath is a 2D mission planner for various types of platforms. Datacx performs two functions:

1. *datacx* the two standard cost functions (temporal std and spatial average) for use within EMPATH.
2. *datacx* computes the 3D (lon,lat,time) depth averaged velocity fields (U(x,y,t) and V(x,y,t)).

Usage

datacx can be run from the command line or a unix-shell with the following syntax:

```
$path/datacx  --option1 value1 -–option2 value2…--option n value n
```

where $path is the path to the directory where the executable datacx resides.

Options

Input ocean forecast filenames:

--path_ncom $filepath/${runname}t%03d.nc

This option directs the code to the directory and filename of the ncom formatted files to be processed. $filepath is the directory of the files, $runname is the defined common netcdf filename and t%03d signals to datacx to loop through all files of the form runnamet000.nc, runnamet003.nc, etc.

Input bathymetry filenames:
--path_etop $filepath/filename.nc

This option directs the code to the directory and filename of the bathymetry file. Bathymetry is used as a mask. If a bathymetry file is not specified, the code computes the cost functions and water velocities where there are acceptable values.

Bathymetry Sign
--bath_sign_multplier -1

The OASIS standard definition of bathymetry is negative downward from the ocean surface. Some bathymetry databases have positive depths below the sea-surface and this can be easily addressed by inputing a bathymetry multiplier of -1.

Number of files:
--ncom_count 25

This option specifies the number of files for *datacx* to sweep through. [The default is all of the files that satisfy the input filename syntax]

Frequency of files:
--ncom_step 6

This option specifies the number of frequency of NCOM files. [The default is 3 hours]

Number of hours:
--ncom_hours 72

If the mamimum number of hours is greater than the files available, *datacx* will loop over the available files. Conversely, if an area analysis wanted to start at a different time than analysis 00, earlier files could be eliminated from the RELO NCOM directory, and *datacx* would grab all of the rest of the files. EMPath will still assume that the run begins at analysis time 00.

Maximum glider depth

--max_depth 1000 [defaults to 1000]

This option specifies the maximum depth of the glider dives. Specifically it sets the zmax for the water velocity average calculation.

4D Currents

--copy_4d_currents

The 4D version of EMPath requires water currents to be specified as 4D fields. With this option, the DATACX code is run as in the 3D case (x,y,t) but the entire water_u and water_v matrices from the NCOM forecast are written to the output nc file in addition to the depth averaged current.

Appendix 3 compilation

Compilation of EMPath uses the form

```
${CC} ${CFLAGS} -c oasis_math.c
        ${CXX} ${CXXFLAGS} -c Genetix.cpp
        ${CXX} ${CXXFLAGS} -c EMPath.cpp
        ${CXX} -o empath oasis_math.o Genetix.o EMPath.o ${LIBS}
```

EMPath is C++, which will vary depending on architecture. Some examples used at the Naval Research Laboratory and Naval Oceanographic Office are below.

On Linux64:

```
CC=gcc
CFLAGS= -03 –Wall -ansi
CXX = g++
CXXFLAGS = -c -O3 -Wall -fpermissive -Wno-write-strings
LIBS = -lpthread -lnetcdf_c++ -L/usr/lib64  -L/common/netcdf/gnu/4.1.2/lib
-lpng
INCPATH = -I/common/netcdf/gnu/4.1.2 /include -I.
```

On the Cray XT5:

```
module swap PrgEnv-pgi PrgEnv-gnu
module load netcdf
module load hdf5
CC=cc
CFLAGS= -O3 –Wall -ansi
CXX = CC
CXXFLAGS = -c -O3
LDFLAGS =
LIBS = -L/opt/cray/xt-sysroot/default/usr/lib64 -lpng -lz
INCPATH = -I.
```

On the IBM P6:

```
CC=xlc
CFLAGS=-q64 -qrtti
CXX = /usr/vacpp/bin/xlc++
CXXFLAGS = -c -qrtti -q64
LDFLAGS = -q64 -qrtti
LIBS = -L/site/libpng-1.2.29_64/lib -L/site/zlib-1.2.3_64/lib -
L/site/netcdf64/lib -lpthread  -lnetcdf_c++ -lnetcdf -lpng -lz
INCPATH = -I/site/libpng-1.2.29_64/include -I/site/zlib-1.2.3_64/include -
I/site/netcdf64/include -I.
```

Datacx is written in C and is compiled as follows:

On Linux 64:

```
gcc -O3 -Wall -L/common/netcdf/gnu/4.1.2/lib -I/common/netcdf/gnu/4.1.2
/include -o datacx datacx.c -lnetcdf -lm
```

On Cray XT5:

```
 module swap PrgEnv-pgi PrgEnv-gnu
 module load netcdf
 module load hdf5
datacx :  datacx.c oasis_ncom_ops.c oasis_ncom_ops.h
 cc -O3 -o datacx datacx.c  oasis_ncom_ops.c -lnetcdf -lhdf5_hl -lhdf5 -lm -lz
```

On IBM P6:

```
xlc -qrtti -q64 -I/site/netcdf64/include -L/site/netcdf64/lib -o datacx datacx.c
oasis_ncom_ops.c -lnetcdf -lm
```